IN THE CLAIMS

Please amend the claims as follows.

1-51 (Cancelled)

52. (Currently Amended) A CPU for executing stack and register-based instructions, comprising:

execute logic for executing the register-based instructions;

a register file associated with the execute logic; and

a hardware accelerator to process the stack-based instructions in cooperation with the execute logic, wherein the hardware accelerator generates a new virtual machine program counter (PC) due to a "jump subroutine JSR" or "jump subroutine wide JSR_W" bytecode by sign extending an immediate branch offset following the "jump subroutine JSR" or "jump subroutine wide JSR_W" bytecode and adding it to a virtual machine program counter (PC) of a current bytecode instruction, computes a return virtual machine program counter and pushes the return virtual machine program counter (PC) onto an operand stack.

53-108. (Cancelled)

56. (Previously Presented) A central processing unit (CPU) for executing stack and register-based instructions, comprising:

execute logic for executing register-based instructions;

a register file associated with the execute logic ; and

a hardware accelerator to process the stack-based instructions in cooperation with the execute logic , wherein the hardware accelerator performs sign extension for virtual machine SiPush and BiPush bytecodes and makes the sign extended data available to be read by the execute logic.

57-125. (Cancelled)

126. (Currently Amended) A central processing unit (CPU) for executing stack and register-based instructions, comprising:

execute logic for executing the register-based instructions ;

a register file associated with the execute logic ; and

a hardware accelerator to process the stack-based instructions in cooperation with the execute logic , wherein the hardware accelerator:

maintains an operand stack for the stack-based instructions in the register file such that the operand stack in the register file defines a ring buffer in conjunction with an overflow/underflow mechanism for moving operands in the operand stack between the register file and memory, and loads variables required for processing the stack-based instructions into the register file;

generates a new virtual machine program counter due to a "GOTO" or "GOTO_W" bytecode by sign extending an immediate branch offset following the "GOTO" or "GOTO_W" bytecode and adds it to a virtual machine program counter of a current bytecode instruction;

generates a new virtual machine program counter due to a Jump sub routine (JSR) or a Jump subroutine wide "JSR_W" bytecode by sign extending an immediate branch offset following the "JSR" or "JSR_W" bytecode and adding it to a virtual machine program counter (PC) of a current bytecode instruction, computes a return virtual machine program counter and pushes the return virtual machine program counter onto the operand stack;

performs a sign extension for virtual machine SiPush and BiPush bytecodes and appends the sign extended data to an immediate field of a register-based instruction being composed based on the stack-based instructions[[,]] ;

performs sign extension for ~~the~~ virtual machine SiPush and BiPush bytecodes and makes the sign extended data available to be read by the execute logic ; and

produces exceptions in respect of selected stack-based instructions.

127. (Previously Presented) The CPU of claim 126, wherein the exceptions are processed using register-based instructions.

128. (Previously Presented) The CPU of claim 126, wherein the hardware accelerator processes the stack-based instructions in cooperation with the execute logic by converting the stack-based instructions into register-based instructions for execution in the execute logic

129-146 (Cancelled)

147. (Original) A central processing unit (CPU) comprising:

execute logic to receive and process input corresponding to register-based instructions;

a hardware accelerator to process stack-based instructions to produce an output that can be processed by the execute logic;

an operand stack for the stack-based instructions, the operand stack being maintained in a register file as a ring buffer;

an overflow/underflow mechanism for moving operands in the operand stack between a register file and a memory, said register file also storing data associated with the register-based instructions;

a bytecode buffer that receives stack-based instructions from the memory; and

an instruction decode unit coupled to the bytecode buffer to decode instructions received from the bytecode buffer and to provide an indication of how many bytes have been processed; and

a common program counter for the stack-based instructions and the register-based instructions, wherein the common program counter is incremented by the indication of the number of bytes processed.

148. (Original) The CPU of claim 147, further comprising a common instruction cache for the stack-based instructions and the register-based instructions.

149. (Original) The CPU of claim 147, wherein the instruction decode unit produces an indication for a variable stored in the register file

150. (Original) The CPU of claim 147, further comprising a microcode unit coupled to the instruction decode unit to receive output therefrom.

151. (Original) The CPU of claim 147, wherein the hardware accelerator produces an exception in respect of selected stack-based instructions.

152. (Original) The CPU of claim 151, wherein the exception is processed using register-based instructions.

153. (Original) The CPU of claim 147, wherein the instruction decode unit decodes multiple instructions received from the bytecode buffer in parallel.

154. (Original) The CPU of claim 150, wherein the instruction decode unit generates a start address for the microcode unit.

155. (Original) The CPU of claim 154, further comprising a mechanism to select the start address or an incremented start address for the microcode unit.

156. (Original The CPU of claim 153, wherein the instruction decode unit comprises multiple decoders.

157. (Original) The CPU of claim 154, wherein the microcode unit produces a stack update indication to cause the instruction decode unit to generate a new start address for the microcode unit.

158-164. (Cancelled)

165. (Currently Amended) A method for a central processing unit (CPU), comprising:

for register-based instructions, processing the register-based instructions in execute logic capable of processing the register-based instructions; and

for stack-based instructions, processing the stack-based instructions in a hardware accelerator into input the execute logic is capable of processing, wherein the hardware accelerator generates a new virtual machine program counter (PC) due to a "jump subroutine JSR" or "jump subroutine wide JSR_W" bytecode by sign extending an immediate branch offset following the "JSR" or "JSR_W" bytecode and adding it to a virtual machine program counter (PC) of a current bytecode instruction, computes a return virtual machine program counter (PC) and pushes the return virtual machine program counter onto an operand stack.

166. (Original) The method of claim 165, wherein processing the stack-based instructions comprises decoding multiple stack-based instructions in an instruction decode unit in parallel.

167. (Original) The method of claim 165, wherein processing the stack-based instructions comprises generating exceptions in respect of selected stack-based instructions.

168. (Original) The method of claim 166, wherein processing the stack-based instructions comprises generating a start address for a microcode unit in the instruction decode unit.

169. (Original) The method of claim 168, processing the stack-based instructions comprises selecting the start address or an incremented start address for the microcode unit.

170. (Original) The method of claim 165, further comprising storing the stack-based instructions and the register-based instructions in a common instruction cache.

171. (Original) The method of claim 165, wherein processing the stack-based instructions comprises producing an indication for a variable stored in the register file.

172-187. (Cancelled)